xUCINET Tools for Working with Two-Mode/Affiliations Data

These notes discuss commands in xUCINET that are useful in working with two-mode network data that record the relationships between two distinct types of entities.

Setting up a Two-Mode Data Project

For the most part, the process for setting up a data project for a two-mode data network is the same as that for one-mode networks, and so you are referred to an earlier set of notes for guidance. There are a couple of nuances, though. To illustrate, I display the `xCreateProject()` command I used to set up the data set on treaties among western hemisphere countries that was used as an in-class example:

```
WHTreaties<-xCreateProject(
  GeneralDescription = "Two Mode Network--Western Hemisphere Treaties",
  NetworkName = "WHTreaties",
  NETFILE1=(file=file.choose()),
  FileType = "csv",
  InFormatType = "AdjMat",
  NetworkDescription = "Membership of countries in Treaties",
  Mode = c("Country","Treaty"),
  Directed = FALSE,
  Loops = FALSE,
  Values = "Binary",
  Class = "matrix",
  References="From Faust CSW chapter, 2005")
```

The most important difference from the one-mode process is seen in the "`Mode=`" argument, which now contains a list containing two "mode" names. The first of these is for the row entities in your two-mode data and the second is for the column entities. The presence of two entries here signals that you are creating an incidence matrix for a two-mode network (you continue to use "`AdjMat`" in argument "`InFormatType`", however).

The above command reads the data from a .csv file. The first few lines of that file look like this:

```
ID,ACS,Aladi,Amazon Pact,Andean Pact,Caricom,Genplacea,Groupofrio,G3,
IDB,Mercosur,NAFTA,OAS,Pariacen,Sanjosegroup,SELA
Argentina,0,1,0,0,0,1,1,0,1,1,0,1,0,0,1
Belize,1,0,0,0,1,0,0,0,1,0,0,1,0,0,1
....
```

The first line (highlighted in blue here) is all on one line (I split it into two only for display purposes in these notes), and contains the ID code followed by the labels for the column nodes. Each line thereafter contains a label for a row node followed by the relationships between that node and all column nodes.

When adding attributes to a two mode data set via `xAddAttributesToProject()`, you must specify the mode to which the attribute(s) pertain in the "`Mode=`" argument. Otherwise, proceed with that as in the one-mode case.

Restructuring Two-Mode Data Projects

It is often convenient or necessary to change a two-mode incidence matrix into another form. This can be accomplished by way of two new functions.

Creating a Bipartite Network

One can convert the incidence matrix for a two-mode network into a one-mode network that includes one row/column for each row and column of the two-mode network; so if there are N rows and M columns in the incidence matrix, this "bipartite" matrix will have N+M rows and columns. In this one-mode network, there are no within-type edges: all relationships among the N row nodes are defined as 0, and likewise all relationships among the M column nodes are defined as 0. One can transform an incidence matrix into this form via the `xTwoModeToBipartite()` function:

```
xTwoModeToBipartite(NET2, First = "RowNodes")
```

Remarks on arguments:

"`NET2`" is an incidence matrix stored either in a xUCINET data project or an R matrix object.

"First" indicates whether the "`RowNodes`" or the "`ColumnNodes`" should be appear first in the bipartite network. It defaults to "`RowNodes`". With "`RowNodes`" here, the row entities in the incidence matrix are in rows 1 to N of the bipartite network, and the column entities in the incidence matrix in rows N+1 to N+M of the bipartite network; with "`ColumnNodes`", the column entities in the incidence matrix appear in rows 1 to M of the bipartite network, while the row entities in the incidence matrix appear in rows M+1 to N+M of the bipartite matrix.

You will almost always wish to assign the result of this (a bipartite matrix) to an R object.

In connection with a bipartite network, it will likely be useful for you to create a vector that identifies the type of node in each row/column of the bipartite network. You can do this via a command like the following:

```
type<-c(rep(1,N),rep(2,M))
```

This creates a vector of N 1s followed by M 2s, and presumes that "`RowNodes`" appear first in the bipartite matrix. (Switch N and M here if you placed "`ColumnNodes`" first.) You can use such a vector to distinguish the types of entities in plots, for example.

You can use a bipartite network as an argument in other xUCINET functions that require a one-mode network as input, but you should exercise caution in interpreting the results of doing so. The fact that the within-mode relationships in a bipartite network are "structural" 0s (i.e., that they cannot assume any other value than 0) will not be recognized by most other functions, and it can affect the results obtained by applying them to such a matrix.

Obtaining Projection Matrices

A different, and often very important, way of creating one-mode networks from two-mode networks is known as "projection." This projects the two mode network into a space involving one mode or the other, losing sight of the other mode. The one-mode network that results is ordinarily valued. The basic idea of projection is to count the number of overlapping affiliations for each pair of entities in the focal mode—that is, the number of indirect ties between them via shared affiliation with an other-mode entity. Variations on this are sometimes introduced (see "`Measure`" argument below), but in the basic form of projection, entities in the focal mode are linked by integer-valued relationships ranging from zero to the number of other-mode entities.

The function `xTwoModeToOneMode()` converts a two-mode network into a one-mode form for one or the other of the types of entities that define the two-mode network. Its syntax is

```
xTwoModeToOneMode(NET2, Measure = "CrossProd")
```

You will usually want to store the result of this function in an R object for subsequent analyses.

Remarks on arguments:

"`NET2`" is an incidence matrix stored either in a xUCINET data project or an R matrix object. By default, this will create a one-mode matrix linking the row entities in the two-mode incidence matrix. If you instead would like the links of column entities with one another, you should apply this command to the transpose, "`t(NET2)`" rather than "`NET2`" itself.

"Measure" governs the cell entries in the projection matrix. It defaults to "`CrossProd`", which takes the cross-product of the entries in a pair of rows in the incidence matrix (or columns, if you transpose it); without transposition, this counts the number of shared affiliations with column entities for each pair of row entities, or equivalently their number of two-step indirect ties via a column entity.

For many purposes this default setting of "`Measure`" works well, but other choices may be made on occasion; be sure you understand these and your reasons for using them before selecting one of these. Those now available in xUCINET (many other possibilities exist) include

"`CrossProdDMin`": the number of common connections of each pair of nodes with other-mode nodes, divided by the maximum number possible given the degree of both nodes. With binary data, for example, the maximum possible cross-product is the minimum of the degrees of the two nodes for which profiles of connections are being compared.

"`ExactMatch`": the number of identical values in each pair of rows/columns (for binary data, counts matching 0s as well as matching 1s)

"`Jaccard`": the number of common connections to other-mode nodes for each pair of nodes, divided by the total number of other-mode nodes to which at least one node in the pair is connected

"`CrossMin`": for each pair of nodes, sums up (across other-mode nodes) the minimum strength of their relationships with each other-mode node

"`Correlation`": the Pearson correlation between entries in each pair of rows of the incidence matrix (or columns, if you transpose it)

"`Covariance`": the covariance between entries in each pair of rows of the incidence matrix (or columns, if you transpose it)

"`Bonacich`" (not included in documentation as of 10/31/22, but is referenced in chapter 13 of Borgatti et al. book, and does operate): yields an adjusted co-occurrence count for each pair of rows/columns, representing an underlying tendency (on a 0 to 1 scale) for pairs of rows/columns to have overlapping affiliations with entities of the other mode, given the overall degrees of the rows/columns. Discussed in more depth in a Bonacich article published in *Sociological Methodology 1972*.

Basic Network Statistics

Many standard network statistics for two-mode networks can be calculated via one-mode functions, some operating on the incidence matrix and others on the bipartite network derived from it. When working with the bipartite network, one must be careful about how these functions treat the structural 0s for within-mode relationships.

Network size (number of row and column entities) may be calculated via the `NROW()` and `NCOL()` functions in base R.

Density can be suitably calculated by applying the `xDensity()` function to the incidence matrix for a two-mode network. Using the bipartite network yields too small a value.

The number of components can be calculated by applying the `xComponents()` function to the incidence matrix for a two-mode network. It produces a useful report giving the number of components, minimum/maximum component size, etc. for the projection matrix for row entities ("Mode A"), the projection matrix for column entities ("Mode B"), and the bipartite network ("Both").

The average geodesic distance separating pairs of nodes can be calculated by obtaining a geodesic distance matrix for the bipartite network via `xGeodesicDistance()` and taking the average of its elements. The averaging needs to exclude the 0 values on the diagonal, and any infinite (`Inf`) values that arise when there is more than one component in the network.

At this point there seems to be no function for calculating a *transitivity* analog for a two-mode network in xUCINET. The notion of transitivity requires redefinition for two-mode data since closed triads are impossible due to the prohibition on within-type relationships.

Visualization

Many of the visualization techniques that we covered earlier can be used in visualizing two-mode networks. One can visualize the incidence matrix directly; here is the `sna` syntax I used to depict the incidence matrix for treaties among Western Hemisphere countries:

```
par(mar = c(0, 0, 0, 0))
gplot(WHTreaties$WHTreaties,
      gmode = "twomode",
      displaylabels = TRUE,
      usearrows=FALSE,
      label.pos=5,
      label.cex=0.8,
      vertex.cex = 0.1,
      label.col=type,
      edge.col="gray65")
```

A few remarks on argument settings here: "gmode" is set to "twomode" for an incidence matrix; you will usually wish to set "usearrows" to FALSE, as affiliation relationships are ordinarily undirected; I found it useful to use a medium-gray edge color.

A key step is to somehow distinguish the type or mode of nodes. I usually do this using a vector "type" like that created above for bipartite networks. Here, I make the distinction via label colors; labels for countries (mode 1) are colored black and those for treaties (mode 2) are colored red. I opted for this because the labels are relatively long but informative here. So I made nodes very small (vertex.cex setting) and placed labels above the nodes (label.pos setting).

You could also use node colors or shapes to distinguish node type. In fact, the gplot() function will do this for you automatically when you set "gmode" to "twomode".

Equivalently, one can visualize the bipartite network. Here is an sna command that accomplishes that:

```
par(mar = c(0, 0, 0, 0))
gplot(WHTreaties.bip,
      gmode = "graph",
      displaylabels = TRUE,
      usearrows=FALSE,
      label.pos=5,
      label.cex=0.8,
      vertex.cex = 0.1,
      label.col=type,
      edge.col="gray65")
```

This yields largely equivalent results here (that is not a coincidence; when "gmode" to is set to "twomode" with an incidence matrix gplot internally creates and then visualizes a bipartite network; differences between the plots obtained in the two ways shown here are attributable to differences the random numbers used in the spring embedder routine. Note that with a bipartite network as input, "gmode" is set to "graph", and that you must use a vector like "type" to distinguish nodes of different entities.

(Notes on visualization of projection matrices to be added)

Centrality

Degree centrality can be calculated separately from the incidence matrix for the two types of entities via xDegreeCentrality(). By default that function will give centrality scores for the row nodes; to obtain them for the column entities, enter the transpose of the incidence matrix as the network in that command. Both raw and normalized degree centralities will be correct if calculated this way.

You can obtain degree centralities for both types of entities at the same time by submitting the bipartite network to `xDegreeCentrality()`. Raw numbers will then be fine, but normalized ones will be too small because the function does not take account of the structural 0 values for within-type relationships, so it assumes that each element in the bipartite matrix could be related to all N+M-1 other nodes, not just to the other-mode nodes..

Betweenness centrality can be calculated by submitting the bipartite network to `xBetweennessCentrality()`. The raw scores will be fine. Normalized ones are problematic, because the standard calculation of the maximum possible betweenness does not take account of the structural 0 values for within-type relationships. A suitable denominator for normalized betweenness is given on p. 256 of the 1997 Borgatti/Everett *Social Networks* article on network analysis for 2-mode data.

Closeness centrality can also be calculated using the bipartite network via `xClosenessCentrality()`. The raw closeness scores are fine, but the normalized ones are incorrect misleading because the standard calculation of the minimum possible closeness does not take the structural 0 values for within-type relationships into account. A suitable denominator for normalized closeness is given on p. 256 of the 1997 Borgatti/Everett *Social Networks* article on network analysis for 2-mode data.

Eigenvector/reflection centrality can be calculated in a manner similar to the hub/authority scores for directed one-mode data. These calculations involve the incidence matrix and its transpose. If `NET2` is the incidence matrix for a two-mode network, eigenvector/reflection centralities for row entities can be calculated using

```
xEigenvectorCentrality(NET2%*%t(NET2))
```

These scores measure the importance of a row entity by the extent to which it is connected to important column entities.

The corresponding scores for column entities can be calculated using

```
xEigenvectorCentrality(t(NET2)%*NET2)
```

These scores measure the importance of a column entity by the extent to which it is connected to important row entities.

Subgroups in Two-Mode Networks

Bi-cliques are analogues to cliques for a two-mode network; they consist of maximal complete segments of an incidence matrix in which all row entities are linked to all column entities. Thy may be enumerated via the `xBiCliques()` function:

```
xBiCliques(NET2, MinA = 3, MinB = 3)
```

In this, `NET2` is an incidence matrix for a two-mode network, `MinA` gives the minimal number of row entities for which bi-cliques are to be reported, and `MinB` gives the minimal number of column entities for which bi-cliques are to be reported.

This function produces an incidence matrix like that reported by `xCliqueMembership()` for one-mode networks. Its rows are all of the entities (both row and column) in the two-mode network. Its columns are bi-cliques. An entry of 1 indicates that an entity is part of a bi-clique.

A "*core/periphery*" analysis of both row and column entities in a two-mode network attempts to separate them into a "core" (interconnected) set and a "periphery" (atomized) set, analogous to what the xCorePeriphery() function does for a one-mode network. The idea is that the core set of row entities is closely linked with the core set of column entities, etc. Its syntax is very simple:

```
xDualCorePeriphery(NET2)
```

The only argument is `NET2`, the incidence matrix for a two-mode network

This function produces an R list containing two vectors (`[1]` and `[2]`), one for the mapping of row entities and the other for the mapping of column entities in the two-mode network; in these, 0s assign an entity to the peripheral block while 1s assign it to the core block. These vectors may be used in `xDensity()` to create a partitioned density matrix that gives a sense of how successful the core/periphery partitioning is.

To do this, if CP is the object storing the results produced by `xDualCorePeriphery()`, you would use a command like

```
xDensity(NET2,ROWS=CP[[1]],COLS=CP[[2]])
```

(take note of the double brackets "[[" in the above.)

This function also produces a couple of plots, one for each type of entity. These have a vertical axis labeled "EVO" and a horizontal axis labeled "Index". These are not documented at this point (11/2/22), so I can't advise on how to interpret them.

One approach to *community detection* for two-mode networks is implemented within xUCINET, in the `xDualLouvainMethod()` function. This applies the one-mode Louvain Method community detection algorithm to **both** of the projection matrices that can be derived from the incidence matrix for a two-mode network. It thereby obtains one partition of the row entities into communities and another partition of the column entities into communities. These partitions can in turn be applied jointly to separate the two-mode network into subsets of row entities and of column entities that are more and less related to one another. The syntax for this function is again very simple:

```
xDualLouvainMethod(NET2)
```

The only argument is `NET2`, the incidence matrix for a two-mode network

Like `xDualCorePeriphery()`, this function produces an R list containing two vectors (`[1]` and `[2]`), one for the mapping of row entities and the other for the mapping of column entities in the two-mode network. These vectors may be used in `xDensity()` to create a partitioned density matrix that gives a sense of how successful the dual community partitioning is.

To do this, if LM is the object storing the results produced by `xDualLouvainMethod()`, you would use a command like

```
xDensity(NET2,ROWS=LM[[1]],COLS=LM[[2]])
```

(take note of the double brackets "[[" in the above.)

This function also produces a couple of plots of the communities found, one for each type of entity.

One might also identify subgroups in two-mode data by way of an extension to the profile similarity approach used in searching for *structurally similar subsets* in one-mode data. Here, one calculates a measure of profile similarity or distance for each pair of entities of one type based on commonalities in their relationships to all entities of the other type; this can be done for both the row entities and the column entities in a two-mode incidence matrix. One then maps the entities into subgroups by way of hierarchical clustering.

This kind of analysis is discussed on pp. 273-277 of Borgatti et al. It involves applying the function `xStructuralEquivalence()` twice, once to the rows of the incidence matrix for the two-mode network, and then again to its columns (by submitting the transpose to the function). The distance/similarity matrices that result are then submitted to `xHierarchicalClustering()`, a mapping for each is selected from among those proposed, and is subsequently used in `xDensity()` to create a partitioned density matrix for the initial two-mode incidence matrix.

The xUCINET software also includes an `xDualStructuralEquivalence()` function. It initiates a related analysis, but instead begins by calculating both projection matrices (for the row and column entities in the incidence matrix) and then calculating Euclidean distances for those. This leads to a related but distinct partitioning from that described above.

The xUCINET documentation describes this function very simply:

```
xDualStructuralEquivalence(NET2)
```

The xUCINET documentation as of 11/2/22 gives only `NET2` (the incidence matrix) as an argument. Although a warning message that appears after the function is executed suggests that all similarity and distance measures that are available for the "`Method=`" argument in `xStructuralEquivalence()` are also available for `xDualStructuralEquivalence()`, this function doesn't accept any of them. It always provides Euclidean distances between the rows of the respective projection matrices.

This function produces an R list containing two matrices (`[1]` and `[2]`), one for the distances of row entities to one another and the other for the distances of column entities to one another. These matrices may be submitted separately to `xHierarchicalClustering()` to obtain proposed mappings of row/column nodes to subgroups; after selecting a mapping for each from among those proposed, those may in turn be used in `xDensity()` to create a partitioned density matrix for the initial two-mode incidence matrix.